投资学五因子案例

林新凯 2016312010173
吕自立 2016302010145
杨宸宇 2016301550186

摘要

本案例研究基于 Fama & French 1993 年在 The Journal of Finance 上发表的论文“Common Risk Factors in the Returns on Stocks and Bonds”，数据来源于万德数据库，调仓频率为月度，检验时间为 2018 年 1 月 1 日至 2017 年 12 月 31 日。其中我们主要考虑的因子包含了原始的 Fama 三因子，即市场风险因子，市值风险因子 SMB，账面市值比风险因子 HML 在内，度量波动的动量因子 MOM，盈利水平风险因子 RMW 与投资水平风险因子 CMA，整体发现因子较为有效，除了动量因子表现稍差意外其余因子都十分显著。

概述

本案例研究基于 Fama & French 1993 年在 The Journal of Finance 上发表的论文“Common Risk Factors in the Returns on Stocks and Bonds”，参考 Fama 的五因子模型以及 Carhart 的四因子模型，数据来源于万德数据库，调仓频率为月度，检验时间为 2018 年 1 月 1 日至 2017 年 12 月 31 日。其中我们主要考虑的因子包含了原始的 Fama 三因子，即市场风险因子，市值风险因子 SMB，账面市值比风险因子 HML 在内，度量波动的动量因子 MOM，盈利水平风险因子 RMW 与投资水平风险因子 CMA。经过研究，我们发现 HML 和 SMB、ROE 和 SMB 等因子间的相关性在研究期间比较高，从整体上来看，调整 R^2 较为理想，平均可以达到 94.53%，除去动量因子意外其他因子的 t 检验均显著，特别是 Market 和 SMB 因子，但在不同的 25 个组合中 β 和 SMB 相差较大。

研究方法

取样本期（2008-2017）的数据按照如下公式进行回归：

$$E_(r_i) - r_f = a_i + b_i[E(r_m) - r_f] + s_iE[SMB] + h_iE[HML] + d_iE[RMW] + c_iE[CMA] + m_iE[MOM]$$

其中，解释变量：

$[E(r_m) - r_f]$：市场指数（万德全 A）相对无风险利率（SHIBOR 隔夜利率）的超额收益

$[SMB]$：每个月按照总市值排列股票池（全部 A 股），依据上下四分位划分大小，即总市值最大的 25% 的股票作为大市值组合，总市值最小的 25% 的股票作为小市值组合，用小市值组合该月的个股收益率平均值减去大市值组合该月的个股收益率平均值，再减去无风险利率得到当月的 SMB 值，并按照这个方法计算样本期内每个月的所有 SMB 值.

$[HML]$：计算方式与 [SMB] 相同，只是将总市值换为市净率，用当月市净率最低（即账面市值比 BM 值最高）的投资组合的个股月度平均收益率减去当月市净率最高的投资组合的个股月度平均收益率，得到当月的 HML 值，再减去无风险利率得到 HML 的超额收益，即当月的 HML 值，并同样按照这个方法计算样本期内每个月的所有 HML 值.

$[RMW]$：计算方式与 [SMB] 相同，只是将总市值换为净资产收益率，用当月净资产收益率最低的投资组合的个股月度平均收益率减去当月净资产收益率最高的投资组合的个股月度平均收益率，得到当月的 RMW 值，再减去无风险利率得到 RMW 的超额收益，即当月的 RMW 值，并同样按照这个方法计算样本期内每个月的所有 RMW 值.

$[CMA]$：计算方式与 [SMB] 相同，只是将总市值换为资产增长率，用当月资产增长率最低的投资组合的个股月度平均收益率减去当月资产增长率最高的投资组合的个股月度平均收益率，得到当月的 CMA 值，再减去无风险利率得到 CMA 的超额收益，即当月的 CMA 值，并同样按照这个方法计算样本期内每个月的所有 CMA 值.

$[MOM]$：首先对前 11 期月度收益率进行排序得到前 30% 和后 30% 股票，计算两类的月度收益率差值，即得当月的 MOM 值，并同样按照这个方法计算样本期内每个月的所有 MOM 值.
被解释变量：

$E_(r_i) - r_f$：依据市值、账面价值/市值两个指标，按照五分位划分得到的 5 个大、中、小投资组合，再排列组合成 25 个交集，得到 25 个投资组合，计算他们的超额收益作为被解释变量.

数据

样本期：2008-01-01 至 2017-12-31

股票池：全部 A 股

剔除所有 ST 股票当月数据

剔除所有非正常交易股票当月数据

投资组合调仓频率：月度（每月末）

市场指数：用万德全 A 指数代替

无风险利率：用 SHIBOR 隔夜拆借利率的月度平均值代替

总市值（ME)：用公司的股权公平市场价值代替

账面市值比（BM)：用市净率代替

账面价值（BE)：用最新报告期资产负债表股东权益代替

净资产收益率：用万德的月度 ROE 代替

资产增长率：用万德的总资产增长率（相较年初）代替

报告内容

In [0]:
```
# Change the file path
import os
try:
        os.chdir(os.path.join(os.getcwd(), '..\OneDrive\工作\学习\第五学期\投资学'))
        print(os.getcwd())
except:
        pass
```

In [34]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
plt.rcParams.update({"font.size": 11})
import numpy as np
import statsmodels.api as sm
```

```
        from IPython.display import display
        import seaborn as sns
```

```
In [35]: path = os.getcwd()
```

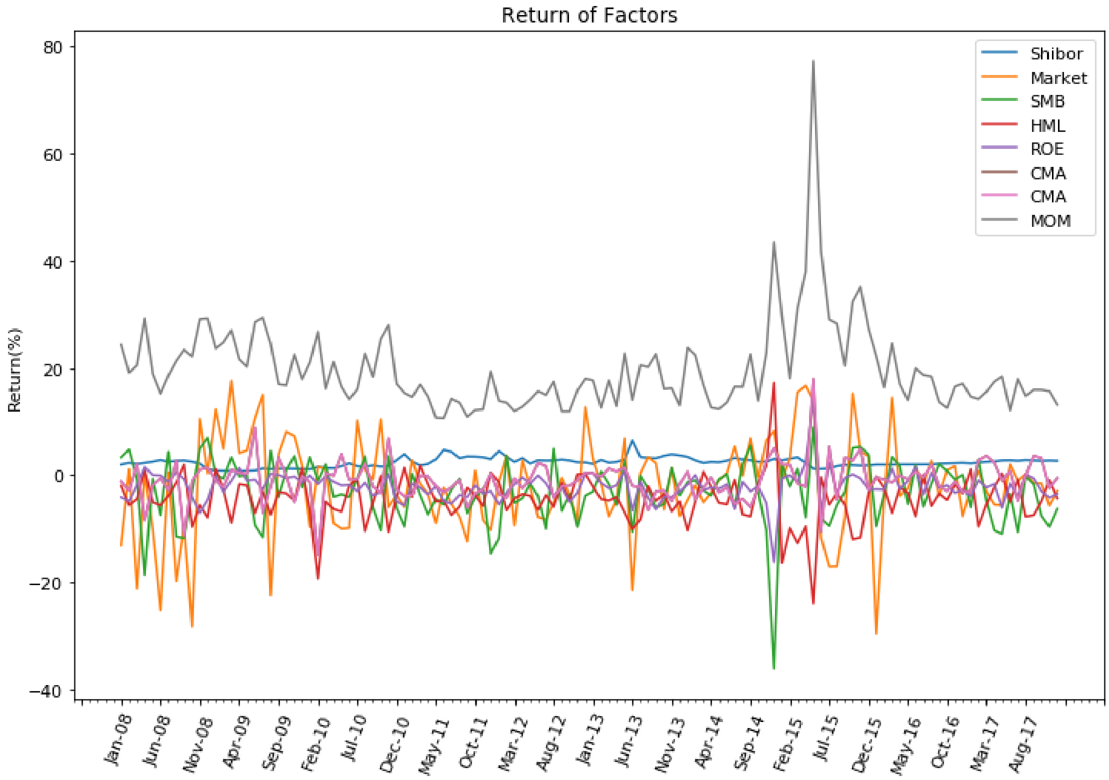
```
In [36]: data = pd.read_csv(
             open(
                 path + "\\FamaFrench.csv",
                 'r',
                 encoding = "utf-8"
             ),
             index_col = [0]
         )
         # data.index = pd.to_datetime(data.index, format = '%b-%y').strftime('%Y-%m')
```

1 收益率

可以看到，动量因子的收益率远高于其它因子，尤其是在 2015 年中。其余因子的收益率表现相差不大，均值都低于 0。

```
In [37]: plt.figure(figsize = (12, 8))
         ax = plt.axes()
         ax.xaxis.set_major_locator(ticker.MultipleLocator(5))
         ax.xaxis.set_minor_locator(ticker.MultipleLocator(1))
         plt.xticks(rotation = 70)
         plt.plot(data.index, data["Shibor"], label = "Shibor")
         plt.plot(data.index, data["Market"], label = "Market")
         plt.plot(data.index, data["SMB"], label = "SMB")
         plt.plot(data.index, data["HML"], label = "HML")
         plt.plot(data.index, data["ROE"], label = "ROE")
         plt.plot(data.index, data["CMA"], label = "CMA")
         plt.plot(data.index, data["CMA"], label = "CMA")
         plt.plot(data.index, data["MOM"], label = "MOM")
         plt.legend()
         plt.ylabel("Return(%)")
         plt.title("Return of Factors")
```

```
Out[37]: Text(0.5,1,'Return of Factors')
```



In [38]: data.describe()

Out[38]:

	Shibor	Market	SMB	HML	ROE	CMA	\
count	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	
mean	2.391529	-1.855738	-3.077592	-4.464530	-2.133120	-0.963855	
std	0.889648	8.989121	5.958622	4.553039	2.819122	3.803802	
min	0.803582	-29.535931	-36.079260	-23.947787	-16.207890	-14.895867	
25%	1.866313	-6.722316	-6.071562	-6.560288	-3.512013	-2.847319	
50%	2.413980	-1.355685	-2.925407	-4.442765	-1.914440	-1.000553	
75%	2.806922	2.844453	0.813304	-1.964928	-0.589128	0.741648	
max	6.468176	17.584159	8.897063	17.213223	14.100332	17.898765	

	MOM	ME0BP0	ME0BP1	ME0BP2	...	ME3BP0	\
count	120.000000	120.000000	120.000000	120.000000	...	120.000000	
mean	19.761993	1.664841	1.117263	0.442514	...	3.702431	
std	8.325334	11.050989	11.104426	10.730139	...	11.721207	
min	10.611562	-26.104895	-31.835864	-33.176358	...	-26.150575	

```
25%     14.612304  -5.393821  -5.205359  -5.765252    ...      -3.062910
50%     17.552181   2.758905   2.282874   1.378960    ...       3.345745
75%     22.609008   8.841230   6.832669   6.706652    ...       9.730519
max     77.189281  31.084765  31.253123  25.181478    ...      63.250741

            ME3BP1      ME3BP2      ME3BP3      ME3BP4      ME4BP0      ME4BP1  \
count  120.000000  120.000000  120.000000  120.000000  120.000000  120.000000
mean     2.362642    1.780316    1.294371    0.762216    4.320273    1.809436
std     10.368887   10.244759   10.125552    9.594999   11.353038    9.085804
min    -26.386979  -29.065758  -29.133660  -26.452483  -21.352238  -23.547693
25%     -3.097955   -3.228491   -3.978126   -4.311305   -1.533493   -3.284972
50%      2.389918    2.167422    1.548886    0.767332    4.003287    2.385493
75%      7.411839    7.401030    6.484346    5.833401    8.323933    6.311672
max     38.905684   31.259587   24.768550   22.834147   62.828819   25.823137

            ME4BP2      ME4BP3      ME4BP4
count  120.000000  120.000000  120.000000
mean     1.265192    1.735443    1.100763
std      9.230414    8.721200    9.092975
min    -25.639417  -22.475910  -26.939990
25%     -2.977532   -2.524531   -3.831161
50%      2.117511    2.054107    1.382359
75%      5.552510    6.588335    5.345399
max     21.750587   22.480232   31.554630

[8 rows x 32 columns]
```

2 统计报告

2.1 因子间相关性

因子间的相关性比较大，这使得模型可能出现过拟合。其中 HML 和 SMB、ROE 和 SMB 等因子间的相关性特别高。如图为 SMB 和 HML 的图例。

```
In [39]: factors = list(data.columns[1:7])
         correlation = pd.DataFrame(index = factors, columns = factors)
         for factor_x in factors:
             for factor_y in factors:
```

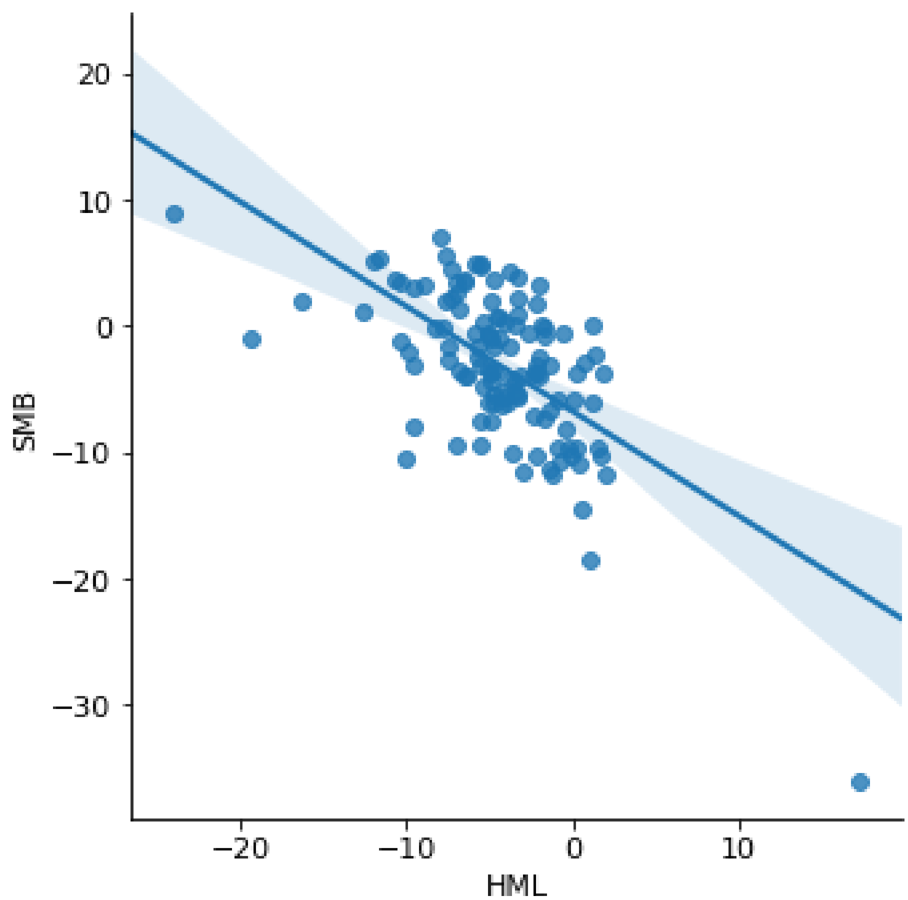
```
        correlation.loc[factor_x, factor_y] = np.corrcoef(
            data[factor_x], data[factor_y]
        )[0][1]
correlation
```

Out[39]:

	Market	SMB	HML	ROE	CMA	MOM
Market	1	0.209565	-0.0927207	0.14921	0.0843781	0.292963
SMB	0.209565	1	-0.634243	0.445343	0.101804	0.0489041
HML	-0.0927207	-0.634243	1	-0.554817	-0.230356	-0.298643
ROE	0.14921	0.445343	-0.554817	1	0.381226	0.312093
CMA	0.0843781	0.101804	-0.230356	0.381226	1	0.325905
MOM	0.292963	0.0489041	-0.298643	0.312093	0.325905	1

In [40]:
```
sns.lmplot("HML", "SMB", data)
```

Out[40]: <seaborn.axisgrid.FacetGrid at 0x1848974aa20>



2.2 因子参数

在 25 个分组的被解释变量中，因子的系数值相差不大，模型对不同风格的投资组合的解释性比较强。

In [41]:
```python
for factor in factors:
    parameters = pd.DataFrame(
        index = ["ME" + str(i) for i in range(5)],
        columns = ["BP" + str(i) for i in range(5)]
    )
    parameters.index.name = "Parameters of " + factor
    for i in range(5):
        for j in range(5):
            y = list(data["ME" + str(i) + "BP" + str(j)])
            x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
            x = sm.add_constant(x)
            result = sm.OLS(y, x).fit()
            parameters.iloc[i, j] = result.params[factor]
    display(parameters)
```

Parameters of Market	BP0	BP1	BP2	BP3	BP4
ME0	0.973124	0.983065	0.951422	0.994599	0.97529
ME1	1.00197	0.955012	0.989937	0.991315	0.960902
ME2	0.937634	0.985608	0.996106	0.965257	1.01143
ME3	0.961281	0.955536	0.994559	1.02404	0.977843
ME4	1.03506	0.989141	1.02331	0.934456	0.989226

Parameters of SMB	BP0	BP1	BP2	BP3	BP4
ME0	0.688744	0.714717	0.770626	0.700231	0.699419
ME1	0.551344	0.61614	0.658785	0.594067	0.616434
ME2	0.46613	0.435887	0.45953	0.529243	0.504404
ME3	0.330681	0.341306	0.361262	0.32556	0.288235
ME4	-0.668081	-0.252607	-0.223159	-0.296529	-0.299492

	BP0	BP1	BP2	BP3	BP4

Parameters of HML

ME0	-0.26278	-0.276935	-0.162308	-0.125571	-0.0648696
ME1	-0.343722	-0.275033	-0.201705	-0.0894505	-0.0357298
ME2	-0.359325	-0.330413	-0.201201	-0.111165	-0.0188457
ME3	-0.464203	-0.343909	-0.183689	-0.159066	0.000528283
ME4	-1.48505	-0.201939	-0.103954	-0.0120891	0.311209

	BP0	BP1	BP2	BP3	BP4
Parameters of ROE					
ME0	-0.285129	-0.225641	-0.293962	-0.214524	-0.286509
ME1	-0.126359	-0.133787	-0.246443	-0.26342	-0.273881
ME2	0.27079	0.0149832	-0.0506243	-0.164069	-0.180784
ME3	0.293409	0.0271374	0.0218246	-0.0803569	-0.0717324
ME4	-0.166685	-0.361995	-0.271425	-0.0192589	-0.221915

	BP0	BP1	BP2	BP3	BP4
Parameters of CMA					
ME0	0.00758401	-0.117193	-0.110249	-0.126299	-0.211194
ME1	0.158363	-0.138066	-0.0895266	-0.114957	-0.212069
ME2	0.213959	-0.123631	-0.145988	-0.226035	-0.222074
ME3	0.0104104	-0.0299374	-0.186399	-0.155059	-0.136125
ME4	-0.420353	0.0984175	0.0822943	-0.139896	-0.123009

	BP0	BP1	BP2	BP3	BP4
Parameters of MOM					
ME0	0.0255467	0.0502839	0.0158153	-0.0395347	-0.0356836
ME1	0.120557	0.0670917	0.0353972	0.0162832	0.000880258
ME2	0.260102	0.104457	0.066073	0.0713211	0.0171894
ME3	0.281403	0.117339	0.078533	0.0117596	0.0509579
ME4	0.105291	0.0407669	-0.0211754	0.103513	0.0329027

2.3 因子 t 值

因子的 t 值指出其统计显著性。大部分的因子都至少在部分投资组合上展现出显著性。尤其是 Market 和 SMB 因子。

```
In [42]: for factor in factors:
             tvalues = pd.DataFrame(
                 index = ["ME" + str(i) for i in range(5)],
                 columns = ["BP" + str(i) for i in range(5)]
             )
             tvalues.index.name = "t values of " + factor
             for i in range(5):
                 for j in range(5):
                     y = list(data["ME" + str(i) + "BP" + str(j)])
                     x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
                     x = sm.add_constant(x)
                     result = sm.OLS(y, x).fit()
                     tvalues.iloc[i, j] = result.tvalues[factor]
             display(tvalues)
```

t values of Market	BP0	BP1	BP2	BP3	BP4
ME0	29.8646	44.6171	42.2722	44.1569	41.61
ME1	31.9636	41.7388	45.8218	42.5176	44.7146
ME2	22.572	34.1663	36.8254	40.2031	46.67
ME3	26.528	33.9254	40.1527	42.5493	39.7577
ME4	21.9876	38.0984	39.3933	39.147	58.2079

t values of SMB	BP0	BP1	BP2	BP3	BP4
ME0	10.939	16.7875	17.7198	16.0888	15.4431
ME1	9.10245	13.9362	15.7812	13.1864	14.8453
ME2	5.80734	7.81989	8.79203	11.4079	12.0451
ME3	4.72276	6.27126	7.54812	7.00067	6.065
ME4	-7.3447	-5.03531	-4.44596	-6.42893	-9.12021

t values of HML	BP0	BP1	BP2	BP3	BP4
ME0	-3.00392	-4.6817	-2.68613	-2.07656	-1.03089
ME1	-4.0843	-4.47736	-3.47767	-1.42905	-0.619308
ME2	-3.22204	-4.26636	-2.77063	-1.72461	-0.323905

```
ME3             -4.77165 -4.54808 -2.76232  -2.46183  0.00800064
ME4             -11.7506 -2.89717 -1.49062 -0.188642     6.82094
```

```
                      BP0       BP1       BP2       BP3       BP4
t values of ROE
ME0             -2.29255  -2.68303  -3.42184  -2.49525   -3.2025
ME1             -1.05608  -1.53191  -2.98861  -2.96002  -3.33903
ME2              1.70788  0.136077 -0.490331  -1.79032  -2.18549
ME3              2.12136  0.252426  0.230844 -0.874756 -0.764109
ME4            -0.927675  -3.65291   -2.7375 -0.211378  -3.42106
```

```
                      BP0       BP1      BP2      BP3      BP4
t values of CMA
ME0              0.09579  -2.18905 -2.01599 -2.30772 -3.70833
ME1              2.07917  -2.48342 -1.70549  -2.0292 -4.06144
ME2              2.11983  -1.76381 -2.22123 -3.87459 -4.21726
ME3             0.118237 -0.437446 -3.09714 -2.65158 -2.27784
ME4             -3.67501    1.5601  1.30383   -2.412 -2.97891
```

```
                     BP0      BP1       BP2       BP3        BP4
t values of MOM
ME0              0.66749  1.94298  0.598244  -1.49434   -1.29614
ME1              3.27428  2.49643   1.39494  0.594588  0.0348739
ME2              5.33091  3.08284   2.07963   2.52904   0.675277
ME3              6.61154  3.54683   2.69934  0.415997    1.76394
ME4              1.90424  1.33683 -0.694017   3.69194    1.64831
```

2.4 因子的 R^2

R^2 统计模型对数据的解释性，可以看到模型的解释性良好，在不同被解释变量中的平均值高达 94.8%。最好的有 97.29%，最差的也有 94.16%。

```
In [43]: rsquared = pd.DataFrame(
             index = ["ME" + str(i) for i in range(5)],
             columns = ["BP" + str(i) for i in range(5)]
```

```
    )
    rsquared.index.name = "R square of Regression"
    for i in range(5):
        for j in range(5):
            y = list(data["ME" + str(i) + "BP" + str(j)])
            x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
            x = sm.add_constant(x)
            result = sm.OLS(y, x).fit()
            rsquared.iloc[i, j] = result.rsquared
    display(rsquared)
```

R square of Regression	BP0	BP1	BP2	BP3	BP4
ME0	0.932553	0.969457	0.965867	0.965743	0.960922
ME1	0.941684	0.964263	0.968883	0.960963	0.964629
ME2	0.905969	0.944476	0.948719	0.956993	0.965179
ME3	0.925854	0.942757	0.954651	0.956172	0.949026
ME4	0.866619	0.936654	0.938558	0.941882	0.972901

```
In [44]: sum(list(rsquared.mean()))/5

Out[44]: 0.9480548778434097

In [45]: max(list(rsquared.max()))

Out[45]: 0.9729007171282669

In [46]: min(list(rsquared.max()))

Out[46]: 0.941683652490355
```

2.5

```
In [47]: rsquared_adj = pd.DataFrame(
             index = ["ME" + str(i) for i in range(5)],
             columns = ["BP" + str(i) for i in range(5)]
         )
         rsquared_adj.index.name = "Adjusted R square of Regression"
         for i in range(5):
```

```python
            for j in range(5):
                y = list(data["ME" + str(i) + "BP" + str(j)])
                x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
                x = sm.add_constant(x)
                result = sm.OLS(y, x).fit()
                rsquared_adj.iloc[i, j] = result.rsquared_adj
        display(rsquared_adj)
```

```
                                 BP0       BP1       BP2       BP3  \
Adjusted R square of Regression
ME0                         0.928972  0.967835  0.964055  0.963924
ME1                         0.938587  0.962365  0.967231   0.95889
ME2                         0.900976  0.941528  0.945996   0.95471
ME3                         0.921917  0.939718  0.952243  0.953845
ME4                         0.859537  0.933291  0.935296  0.938796

                                 BP4
Adjusted R square of Regression
ME0                         0.958847
ME1                          0.96275
ME2                          0.96333
ME3                         0.946319
ME4                         0.971462
```

```
In [48]: sum(list(rsquared_adj.mean()))/5

Out[48]: 0.9452967297642989

In [49]: max(list(rsquared_adj.max()))

Out[49]: 0.9714618171527766

In [50]: min(list(rsquared_adj.max()))

Out[50]: 0.9385872092597544
```

2.6 回归报告

取其中一个被解释变量展示回归结果。可以看到，除了动量因子以外的所有银子，在统计上都是显著的 (t>2)。JB 检验和 DW 检验的效果也十分好。

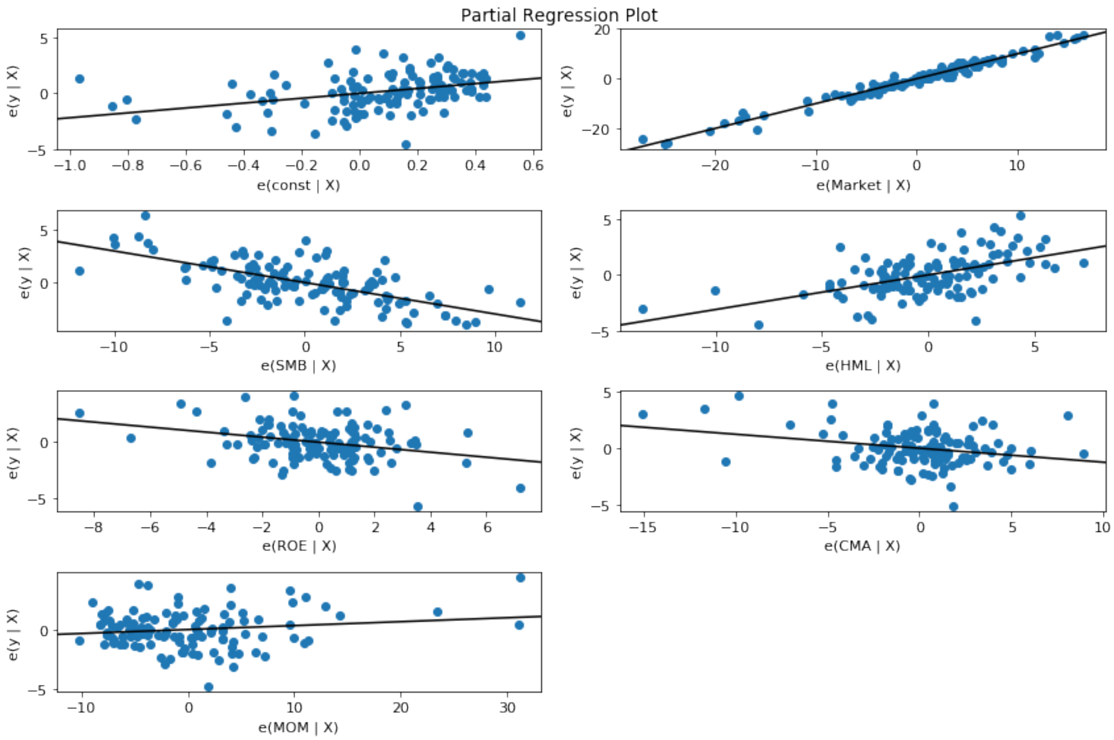
```
In [51]: y = list(data["ME4BP4"])
         x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
         x = sm.add_constant(x)
         result = sm.OLS(y, x).fit()
         print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.973
Model:                            OLS   Adj. R-squared:                  0.971
Method:                 Least Squares   F-statistic:                     676.1
Date:                Wed, 19 Dec 2018   Prob (F-statistic):           4.62e-86
Time:                        21:02:59   Log-Likelihood:                -218.18
No. Observations:                 120   AIC:                             450.4
Df Residuals:                     113   BIC:                             469.9
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.1620      0.489      4.418      0.000       1.192       3.132
Market         0.9892      0.017     58.208      0.000       0.956       1.023
SMB           -0.2995      0.033     -9.120      0.000      -0.365      -0.234
HML            0.3112      0.046      6.821      0.000       0.221       0.402
ROE           -0.2219      0.065     -3.421      0.001      -0.350      -0.093
CMA           -0.1230      0.041     -2.979      0.004      -0.205      -0.041
MOM            0.0329      0.020      1.648      0.102      -0.007       0.072
==============================================================================
Omnibus:                        2.318   Durbin-Watson:                   2.234
Prob(Omnibus):                  0.314   Jarque-Bera (JB):                2.018
Skew:                           0.083   Prob(JB):                        0.365
Kurtosis:                       3.613   Cond. No.                         77.8
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

2.7 因子回归图

In [52]:
```
fig = plt.figure(figsize = (12, 8))
fig = sm.graphics.plot_partregress_grid(result, fig = fig)
```



总结

经过研究，在 2007 年 1 月 1 日到 2017 年 12 月 31 日所有 A 股除去 ST 和非政策交易股票的数据集中，我们发现 HML 和 SMB、ROE 和 SMB 等因子间的相关性在研究期间比较高，从整体上来看，通过 R^2 观察统计模型对数据的解释性，可以看到模型的解释性良好，在不同被解释变量中的 R^2 平均值高达 94.8%。最好的有 97.29%，最差的也有 94.16%。调整 R^2 也较为理想，平均可以达到 94.53%，除去动量因子意外其他因子的 t 检验均显著，特别是 Market 和 SMB 因子，JB 检验和 DW 检验的效果也十分好。但在不同的 25 个组合中 β 和 SMB 相差较大。

参考文献

[1]Common Risk Factors in The Returns On Stocks and Bonds, Eugene F. Fama a & Kenneth R. French, Journal of Financial Economics 33 (1993) 3-56. North-Holland.

[2]On Persistence in Mutual Fund Performances, Carhart, The Journey of Finance, NO.1, March 1997.

[3]A Five-Factor Asset Pricing Model, Eugene F. Fama a & Kenneth R. French, Journal of Financial Economics 116 (2015) 1–22.

[4]Anomalies in Chinese A Shares, Jason Hsu & Vivek Viswanathan & Michael Wang & Phillip Wool.

[5]Size, Value, and Momentum in International Stock Returns, Fama& Eugene F.& Kenneth R. French., Journal of Financial Economics 105, no. 3 (2012): 457-472.

[6]International Tests of A Five-Factor Asset Pricing Model, Fama& Eugene F.& Kenneth R. French.

[7]Profitability, Investment and Average Returns, Fama & Eugene F.& Kenneth R. French., Journal of Financial Economics 82, no. 3 (2006): 491-518.

数据处理部分代码 (Python)

```
In [1]: import os
    import pandas as pd
    PYk+knimport datetime as dt
    from scipy import stats
    import statsmodels.api as sm
    import matplotlib.pyplot as plt
    import seaborn as sns

    import WindPy as w
    from WindPy import *
    w.start()

    path = os.getcwd()
    start = "2008-01-01"
    end = "2017-12-31"

    def months_list(start = start, end = end):
        '''
        参数:
            start: 开始日期 ("YYYY-MM-DD")。(str)
            end: 结束日期 ("YYYY-MM-DD")。(str)
        返回:
            样本期的月份列表。(string list)
        '''
        file_path = path + r"/months.csv"
        if os.path.isfile(file_path):
            months = pd.read_csv(
                open(file_path, 'r', encoding = 'utf-8'),
                index_col = [0]
            )[12:]
            months_list = list(months["Month"])
            months_list = [x[:7] for x in months_list]
            else:
            months = w.tdays(start, end, "Period=M", usedf = True)[1]
            months.columns = ["Month"]
```

```python
        months.to_csv(file_path)
        months_list = list(months["Month"])
        months_list = [x.strftime('%Y-%m') for x in months_list]
    return months_list

months_list = months_list()

def market(
    start = start,
    end = end,
    hs300 = False,
    windA = True,
):
    '''
    参数:
        start: 开始日期 ("YYYY-MM-DD")。(str)
        end: 结束日期 ("YYYY-MM-DD")。(str)
        hs300: 是否用沪深 300 代表市场指数。(bool)
        windA: 用万德全 A 代表市场指数。(bool)
    返回:
        市场指数在样本期内的表现。(pd.DataFrame)
        index: Month, 日期 (YYYY-MM-DD)。(string)
        column: Market, 当月涨跌幅 (%)。(float)
    '''
    file_path = path + r"/market.csv"
    if os.path.isfile(file_path):
        market = pd.read_csv(
            open(file_path, 'r', encoding = 'utf-8'),
            index_col = [0]
        )[12:]
    else:
        if hs300:
            market_code = "000300.SH"
        elif windA:
            market_code = "881001.WI"
        market = w.wsd(
            market_code,
```

```python
            "pct_chg",
            start,
            end,
            "Period=M",
            usedf = True
        )[1]
        market.index = pd.to_datetime(market.index).strftime("%Y-%m")
        market.index.name = "Month"
        market.columns = ["Market"]
        market.dropna(inplace = True) # 剔除缺失值
        market.to_csv(file_path)
    return market

market = market()

def shibor(start = start, end = end):
    '''
    参数:
        start: 开始日期 ("YYYY-MM-DD")。(str)
        end: 结束日期 ("YYYY-MM-DD")。(str)
    返回:
        无风险利率，以 SHIBOR 隔夜利率的月平均值代表。(pd.DataFrame)
        index: Month, 日期 (YYYY-MM-DD)。(pd.datetime)
        column: Shibor, 当月涨跌幅 (%)。(float)
    '''
    file_path = path + r"/shibor.csv"
    if os.path.isfile(file_path):
        shibor = pd.read_csv(
            open(file_path, 'r', encoding = 'utf-8'),
            index_col = [0]
        )[12:]
    else:
        shibor = w.wsd(
            "SHIBORON.IR",
            "close",
            start,
            end,
```

```python
                "",
                usedf = True
            )[1]
            shibor.index = pd.to_datetime(shibor.index).strftime('%Y-%m')
            shibor = pd.DataFrame(shibor.groupby(shibor.index)["CLOSE"].mean())
            shibor.index.name = "Month"
            shibor.columns = ["Shibor"]
            shibor.dropna(inplace = True) # 剔除缺失值
            shibor.to_csv(file_path)
    return shibor

shibor = shibor()

def all_data(
    start = start,
    end = end,
    universe = "A",
    trading_only = True,
    non_ST_only = True
):
    '''
    参数:
        start: 开始日期 ("YYYY-MM-DD")。(str)
        end: 结束日期 ("YYYY-MM-DD")。(str)
        universe: 股票池，沪深 300('hs300') 或全部 A 股 ('A')。(str)
        trading_only: 是否只保留正常交易的股票。(bool)
        non_ST_only: 是否只保留非 ST 股票。(bool)
    返回:
        指定样本期的全部数据。(pd.DataFrame)
        index: Month, 日期 (YYYY-MM)。(str)
        columns:
            Code, 股票代码。(str)
            Name, 股票简称。(str)
            Return, 当月涨跌幅 (%)。(float)
            ME, 当月总市值。(float)
            Book, 当月账面价值。(float)
            Price, 当月股价。(float)
```

```
        BP, 当月账面市值比。(float)
        Asset, 当月账面价值。(float)
        ROE, 权益回报。(float)
        ST, 是否为 ST 股票。(str)
'''

file_path = path + r"/data.csv"

if os.path.isfile(file_path):
    data = pd.read_csv(
        open(
            file_path,
            'r',
            encoding = 'utf-8'
        ),
        index_col = [0]
    )
    if trading_only:
        data = data.dropna() # 剔除缺失值
    if non_ST_only:
        data = data[data['ST'] == '否'] # 剔除 ST 股票
    data["Asset"] = data["Asset"].astype("str")
    data["Asset"] = [''.join(x.split(",")) for x in list(data["Asset"])]
    data["Asset"] = data["Asset"].astype("float")

else:

    if universe == "hs300":
        stocks_list = list(w.wset(
            "sectorconstituent",
            "date="+end+";windcode=000300.SH",
            usedf = True
        )[1].sample(100)['wind_code']) # ".sample(100)" 仅测试用

    elif universe == "A":
        stocks_list = list(
            w.wset(
```

```python
                    "sectorconstituent",
                    "date="+end+";sectorid=a001010100000000",
                    usedf = True
                )[1]['wind_code'] # ".sample(100)" 仅测试用
            )

        data = pd.DataFrame()

        for stock in stocks_list:
            stock_data = w.wsd(
                stock,
                '''trade_code,pct_chg,ev,roe,yoyassets,trade_status
,riskwarning''',
                start,
                end,
                '''unit=1;ruleType=3;period=2;returnType=1;index=000001.SH;
Period=M;Fill=Previous''',
                usedf = True
            )[1]
            stock_data.index = pd.to_datetime(stock_data.index)\
                                    .strftime("%Y-%m")
            data = data.append(stock_data)

        data.index.name = "Month"
        data.columns = [
            "Code", "Return",
            "ME", "ROE",
            "Asset", "Status", "ST"
        ]

        data.to_csv(file_path)

    return data

data = all_data()

def excess(data):
```

```
    '''
    参数:
        data: 要操作的数据表。(pd.DataFrame)
    返回:
        添加了无风险利率和超额收益列的原数据表。(pd.DataFrame)
    '''
    col_list = list(data.columns)
    data["Shibor"] = list(shibor["Shibor"])
    for column in col_list:
        data[column] = data[column] - data["Shibor"]
    return data[col_list]

def value_weighted_data(data):
    '''
    参数:
        data: 要操作的数据表。(pd.DataFrame)
    返回:
        将 Return 列替换为按 ME 列（市值）加权后的 Return。(pd.DataFrame)
    '''
    total_ME = data.sum(axis = 0)["ME"]
    data["Weight"] = data["ME"] / total_ME
    data["Return"] = data["Return"] * data["Weight"]
    return data

def monthly_return(factor, value_weighted = True):
    '''
    参数:
        factor: 因子指标名。(str)
        value_weighted: 是否将收益市值加权。(bool)
    返回:
        每个月按照因子排列的大小投资组合的收益。(pd.DataFrame)
    '''
    small_ret_list, big_ret_list = [], []
    for month, monthly_data in data.groupby(data.index):
        sort = monthly_data.sort_values(by = factor)
        small = sort[:round(len(monthly_data)/3)]
        big = sort[-round(len(monthly_data)/3):]
```

```python
        if value_weighted:
            small_ret = value_weighted_data(small).sum(axis = 0)["Return"]
            big_ret = value_weighted_data(big).sum(axis = 0)["Return"]
        else:
            small_ret = small.sum(axis = 0)["Return"]/len(small)
            big_ret = big.sum(axis = 0)["Return"]/len(big)
        small_ret_list.append(small_ret)
        big_ret_list.append(big_ret)
    monthly_return = pd.DataFrame(index = months_list)
    monthly_return["Small " + factor] = small_ret_list
    monthly_return["Big " + factor] = big_ret_list
    return monthly_return

def MKT(excess_return = True):
    '''
    参数:
        excess_return: 是否计算超额收益。(bool)
    返回:
        指定 Fama French 三因素模型的市场因子。(pd.DataFrame)
    '''
    MKT = market
    MKT.columns = ["MKT"]
    if excess_return:
        MKT = excess(MKT)
    return MKT

MKT = MKT()

def SMB(excess_return = True):
    '''
    参数:
        excess_return: 是否计算超额收益。(bool)
    返回:
        指定 Fama French 三因素模型中的 HML 因子。(pd.DataFrame)
    '''
    data = monthly_return('ME')
    data["SMB"] = data["Small ME"] - data["Big ME"]
```

```python
    if excess_return:
        data = excess(data)
    return data[["SMB"]]

SMB = pd.read_csv(path + r"/SMB.csv", index_col = [0])

SMB.to_csv(path + r"SMB.csv")

def HML(excess_return = True):
    '''
    参数:
        excess_return: 是否计算超额收益。(bool)
    返回:
        指定 Fama French 三因素模型中的 HML 因子。(pd.DataFrame)
    '''
    data = monthly_return('BP')
    data["HML"] = data["Big BP"] - data["Small BP"]
    if excess_return:
        data = excess(data)
    return data[["HML"]]

HML = HML()

def ROE(excess_return = True):
    '''
    参数:
        excess_return: 是否计算超额收益。(bool)
    返回:
        指定 Fama French 模型中的 ROE 因子。(pd.DataFrame)
    '''
    data = monthly_return("ROE")
    data["ROE"] = data["Big ROE"] - data["Small ROE"]
    if excess_return:
        data = excess(data)
    return data[["ROE"]]

ROE = ROE()
```

```python
def CMA(excess_return = True):
    '''
    参数:
        excess_return: 是否计算超额收益。(bool)
    返回:
        指定 Fama French 模型中的 CMA 因子。(pd.DataFrame)
    '''
    data = monthly_return("Asset")
    data["CMA"] = data["Big Asset"] - data["Small Asset"]
    if excess_return:
        data = excess(data)
    return data[["CMA"]]

CMA = CMA()

def MOM(excess_return = True):
    '''
    参数:
        excess_return: 是否计算超额收益。(bool)
    返回:
        指定 Fama French 模型中的 MOM 因子。(pd.DataFrame)
    '''
    data = monthly_return("Return")
    data["MOM"] = data["Big Return"] - data["Small Return"]
    if excess_return:
        data = excess(data)
    return data [["MOM"]]

MOM = MOM()

def Y(excess_return = True):
    '''
    参数:
        excess_return: 是否计算超额收益。(bool)
    返回:
        被解释变量。(pd.DataFrame)
```

```
    '''
    Y = {}
    for i in range(5):
        for j in range(5):
            Y["ME" + str(i) + "BP" + str(j)] = []
    for month, monthly_data in data.groupby(data.index):
        sort_ME = monthly_data.sort_values(by = "ME")
        sort_BP_ME = monthly_data.sort_values(by = "BP")
        length = round(len(monthly_data)/5)
        for i in range(5):
            for j in range(5):
                ME_list = list(sort_ME[i*length:(i+1)*length]["Code"])
                BP_ME_list = list(sort_BP_ME[j*length:(j+1)*length]["Code"])
                stock_list = [x for x in ME_list if x in BP_ME_list]
                portfolio = monthly_data[monthly_data["Code"]\
                                                     .isin(stock_list)]
                portfolio_ret = value_weighted_data(portfolio)\
                                                .sum(axis = 0)["Return"]
                Y["ME" + str(i) + "BP" + str(j)].append(portfolio_ret)
    return pd.DataFrame(Y, index = months_list)

Y = Y()

FamaFrench = pd.DataFrame()
FamaFrench["Shibor"] = shibor["Shibor"]
FamaFrench["Market"] = MKT["MKT"]
#FamaFrench["SMB"] = SMB["SMB"]
FamaFrench["HML"] = HML["HML"]
FamaFrench["ROE"] = ROE["ROE"]
FamaFrench["CMA"] = CMA["CMA"]
FamaFrench["MOM"] = MOM["MOM"]
FamaFrench = pd.concat(
    [FamaFrench, Y],
    axis = 1,
    sort = False
)
```

```
FamaFrench.to_csv(path + r"/FamaFrench.csv")


FamaFrench = pd.read_csv(
    open(
        path + r"/FamaFrench.csv",
        'r',
        encoding = 'utf-8'
    ),
    index_col = [0]
)
```

数据分析部分代码 (Python)

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
plt.rcParams.update({"font.size": 11})
import numpy as np
import statsmodels.api as sm
from IPython.display import display
import seaborn as sns

path = os.getcwd()

data = pd.read_csv(
    open(
        path + "\\FamaFrench.csv",
        'r',
        encoding = "utf-8"
    ),
    index_col = [0]
)
data.index = pd.to_datetime(data.index, format = '%b-%y').strftime('%Y-%m')


# 收益率
# 可以看到，动量因子的收益率远高于其它因子，尤其是在 2015 年中。
plt.figure(figsize = (12, 8))
ax = plt.axes()
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))
ax.xaxis.set_minor_locator(ticker.MultipleLocator(1))
plt.xticks(rotation = 70)
plt.plot(data.index, data["Shibor"], label = "Shibor")
plt.plot(data.index, data["Market"], label = "Market")
plt.plot(data.index, data["SMB"], label = "SMB")
plt.plot(data.index, data["HML"], label = "HML")
plt.plot(data.index, data["ROE"], label = "ROE")
```

```python
plt.plot(data.index, data["CMA"], label = "CMA")
plt.plot(data.index, data["CMA"], label = "CMA")
plt.plot(data.index, data["MOM"], label = "MOM")
plt.legend()
plt.ylabel("Return(%)")
plt.title("Return of Factors")


data.describe()


# 统计报告

# 因子间相关性


factors = list(data.columns[1:7])
correlation = pd.DataFrame(index = factors, columns = factors)
for factor_x in factors:
    for factor_y in factors:
        correlation.loc[factor_x, factor_y] = np.corrcoef(
            data[factor_x], data[factor_y]
        )[0][1]
correlation

sns.lmplot("HML", "SMB", data)


# 因子参数
# 其中系数相对较大的有 $\beta$ 和 SMB。

for factor in factors:
    parameters = pd.DataFrame(
        index = ["ME" + str(i) for i in range(5)],
        columns = ["BP" + str(i) for i in range(5)]
    )
    parameters.index.name = "Parameters of " + factor
```

```python
    for i in range(5):
        for j in range(5):
            y = list(data["ME" + str(i) + "BP" + str(j)])
            x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
            x = sm.add_constant(x)
            result = sm.OLS(y, x).fit()
            parameters.iloc[i, j] = result.params[factor]
    display(parameters)
```

```python
for factor in factors:
    tvalues = pd.DataFrame(
        index = ["ME" + str(i) for i in range(5)],
        columns = ["BP" + str(i) for i in range(5)]
    )
    tvalues.index.name = "t values of " + factor
    for i in range(5):
        for j in range(5):
            y = list(data["ME" + str(i) + "BP" + str(j)])
            x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
            x = sm.add_constant(x)
            result = sm.OLS(y, x).fit()
            tvalues.iloc[i, j] = result.tvalues[factor]
    display(tvalues)
```

```python
rsquared = pd.DataFrame(
    index = ["ME" + str(i) for i in range(5)],
    columns = ["BP" + str(i) for i in range(5)]
)
rsquared.index.name = "R square of Regression"
for i in range(5):
    for j in range(5):
        y = list(data["ME" + str(i) + "BP" + str(j)])
        x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
        x = sm.add_constant(x)
```

```python
        result = sm.OLS(y, x).fit()
        rsquared.iloc[i, j] = result.rsquared
display(rsquared)

rsquared_adj = pd.DataFrame(
    index = ["ME" + str(i) for i in range(5)],
    columns = ["BP" + str(i) for i in range(5)]
)
rsquared_adj.index.name = "Adjusted R square of Regression"
for i in range(5):
    for j in range(5):
        y = list(data["ME" + str(i) + "BP" + str(j)])
        x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
        x = sm.add_constant(x)
        result = sm.OLS(y, x).fit()
        rsquared_adj.iloc[i, j] = result.rsquared_adj
display(rsquared_adj)

# 回归报告
# 取其中一个被解释变量展示回归结果。


y = list(data["ME4BP0"])
x = data.loc[:, ["Market", "SMB", "HML", "ROE", "CMA", "MOM"]]
x = sm.add_constant(x)
result = sm.OLS(y, x).fit()
print(result.summary())

# 因子回归图
fig = plt.figure(figsize = (12, 8))
fig = sm.graphics.plot_partregress_grid(result, fig = fig)
```